02

Clase abstracta

Transcripción

[00:00] En esta clase vamos a ver un concepto ya diferente. Ya conocemos lo que es herencia, lo que es polimorfismo, y vamos a volver a dar un vistazo rápido a cómo es nuestra estructura de clases. Entonces sabemos que gerente extiende de funcionario, y así como el gerente extiende de él, tenemos otras clases también que extienden de funcionario como por ejemplo el contador y bueno, contador, yo pensaba que habíamos creado otra clase, pero bueno.

[00:40] Contador también extiende de funcionario. Ahora, en toda empresa existen diferente cargos. Tenemos diseñadores, contadores, abogados, etcétera. Sabemos que lo que queremos hacer con el software es abstraer esa característica digamos del mundo real y llevarla a un sistema, a una realidad ya sistematizada, ya en código, todo eso.

[01:15] Y el software debería representar lo que nuestro negocio es en realidad, esa es la primera primicia. El software representa lo que es nuestro negocio pero ya de una forma digitalizada y lógica con la cual podemos crear métodos y relaciones entre los diferentes objetos. Primera premisa.

[01:40] Ahora, si bien hay contadores, abogados, todo eso, ¿qué es un funcionario? Funcionario se le llama a cualquier trabajador, sea el cargo que tengas, todos somos funcionarios. Entonces, funcionario no es el cargo que tú le das a alguien dentro de la empresa. Funcionario es el nombre que recibimos todos los que somos, valga la redundancia, funcionarios dentro de la empresa.

[02:19] Entonces funcionario es un término muy, muy general. ¿Cuál es el detalle de aquí? En nuestra implementación, funcionario es nuestra base para saber que yo pertenezco a esta empresa y que tengo estos campos, estas características. Pero funcionario no describe un cargo en la empresa, que es lo que yo estoy representando pues con contador y con gerente.

[02:48] Entonces funcionario no es un concepto de clase. Funcionario lo vamos a ver ahora como un concepto abstracto. Y ya que es un concepto abstracto, en Java tenemos una palabra reservada para este tipo de objetos, que es abstract. "Ctrl + espacio" abstract, aquí está.

[03:16] Abstract nos indica que esta clase de aquí es abstracta. ¿Qué significa que una clase sea abstracta? Vemos que aquí en funcionario el código sigue compilando, no tenemos mayor diferencia, no hay ningún tipo de error, pero vamos a ver este lado de aquí. Vemos que aquí TestControlBonificacion y TestFuncionario ya no están compilando. ¿Por qué es? vamos a TestFuncionario.

[03:52] Y vemos que aquí él me está dando un error en new Funcionario. Aquí comienza lo interesante. ¿Qué es lo que pasa? Que la clase, al ser abstracta, no puede ser instanciada como tal. ¿Por qué? Porque al ser abstracta es una representación conceptual pero no puede ser una representación física del objeto. Entonces, para ello, él me dice: "Okay, tú puedes dar a un funcionario, perfecto, pero ese funcionario es una representación abstracta".

[04:28] Si quieres una instancia, tienes que darme una representación física o real del objeto. Y en este caso pues vamos a decirle que Diego es un contador, solamente para el ejemplo. Si acá pongo contador, que extiende del funcionario, vemos que el código ya compila correctamente y ya no tengo ningún tipo de error. Perfecto. Guardo acá y desapareció el error.

[04:58] ¿Por qué? Porque contador ya es un tipo de clase normal, un objeto propio, es una representación ya física del objeto, extiende una clase abstracta

que es funcionario. Funcionario no puede ser instanciado pero contador sí. Esa es la primera ventaja que tenemos con las clases abstractas.

[05:22] Vamos a ver aquí el segundo que es controlBonificacion, que tampoco está compilando por el mismo error, porque el funcionario Diego está instanciando una clase tipo funcionario que es abstracta. Para esto reemplazamos con contador nuevamente y el código sigue compilando. En el caso de aquí, pues instancias directamente contador, Alexiz = new contador.

[05:49] Vemos que el código está compilando normalmente porque es clase física digamos. Clase normal con clase normal, todo bien. En el caso de la clase abstracta, no. Y vemos que el código ya compila todo correctamente y nosotros ya le dimos una nueva característica a funcionario. Ahora, esos no son todos los beneficios de volver una clase abstracta. En el siguiente video vamos a ver y profundizar un poco más qué estado todo esto de la abstracción.